27장. 배열

27.1 배열이란?

1. 배열(array)은 여러 개의 값을 순차적으로 나열한 자료구조다.

2. 배열이 가지고 있는 값을 **요소**(element),
   요소 자신의 위치를 나타내는 0 이상의 정수인 **인덱스**(index)를 갖는다.

3. 자바스크립트에 배열이라는 타입은 존재하지 않는다. 배열은 객체 타입이다.

```
typeof arr   // object
```

4. 배열과 일반 객체

Aa 구분	객체	배열
구조	프로퍼티 키와 프로퍼티 값	인덱스와 요소
값의 참조	프로퍼티 키	인덱스
값의 순서	X	O
length 프로퍼티	X	O

- 둘의 가장 명확한 차이는 “값의 순서”, “length 프로퍼티”

27.2 자바스크립트 배열은 배열이 아니다

1. **밀집 배열**(dense array)
   : 배열의 요소는 하나의 데이터 타입으로 통일되어 있으며 서로 연속적으로 인접해 있다.

2. 인덱스를 통해 단 한번의 연산으로 임의의 요소에 접근
   : 임의 접근(random access), 시간 복잡도 O(1)

정렬되지 않은 배열에서 특정한 요소를 검색하는 경우,
특정 요소를 발견할 때까지 처음부터 차례대로 검색
: 선형 검색(linear search), 시간 복잡도 O(n)

3. 배열의 요소를 위한 각각 메모리 공간은 동일한 크기 아니어도 되며,
   연속적으로 이어져 있지 않을 수도 있음.
   : **희소 배열**(sparse array)
   - **자바스크립트 배열은 일반적인 배열의 동작을 흉내 낸 특수한 객체다.**

4. 자바스크립트에서 배열이 일반 객체보다 약 2배 정도 빠르다.

27.3 length 프로퍼티와 희소 배열

1. 현재 length 프로퍼티 값보다 작은 숫자 값을 할당하면 배열의 길이가 줄어든다.

2. 현재 length 값보다 더 큰 숫자 값을 할당하는 경우, length 프로퍼티 값은 변경되지만
   실제로 배열의 길이가 늘어나지는 않는다.

3. 이처럼 배열 요소가 연속적으로 위치하지 않고 일부가 비어 있는 배열을 **희소 배열**이라 함.

4. **희소 배열은 length와 배열 요소의 개수가 일치하지 않는다.**
   희소 배열의 length는 희소 배열의 실제 요소 개수보다 언제나 크다.

27.4 배열 생성

27.4.1 배열 리터럴

1. 가장 일반적이고 간편한 배열 생성 방식이다.

27.4.2 Array 생성자 함수

1. Array 생성자 함수를 통해 배열을 생성한다.
   전달된 인수의 개수에 따라 다르게 동작하므로 주의가 필요.

2. new 연산자와 함께 호출하지 않더라도, 즉 일반 함수로서 호출해도

   배열을 생성하는 생성자 함수로 동작한다.

   이는 생성자 함수 내부에서 new.target을 확인하기 때문.

```
Array(1, 2, 3);   // [ 1, 2, 3 ]
```

27.4.3 Array.of

1. 전달된 인수를 요소를 갖는 배열을 생성

27.4.4 Array.from

1. 유사 배열 객체 or 이터러블 객체를 인수로 전달받아 배열로 변환하여 반환한다.

```
// 유사 배열 객체를 변환하여 배열을 생성
Array.from({ length: 2, 0 : 'a', 1 : 'b' });  // ['a', 'b']

// 이터러블을 변환하여 배열을 생성. 문자열은 이터러블이다.
Array.from('Hello');  // ['H', 'e', 'l', 'l', 'o']
```

27.5 배열 요소의 참조

1. 존재하지 않는 요소에 접근하면 undefined가 반환

27.6 배열 요소의 추가와 갱신

1. 존재하지 않는 인덱스를 사용해 값을 할당하면 새로운 요소가 추가된다.

   이때 length 프로퍼티 값은 자동 갱신됨.

2. 정수 이외의 값을 인덱스처럼 사용하면 요소가 생성되는 것이 아니라, 프로퍼티가 생성됨.

27.7 배열 요소의 삭제

1. 배열은 사실 객체이기 때문에 delete 연산자 사용 가능

2. 희소 배열을 만들게 될 수 있으므로 delete 연산자는 사용하지 않는 것이 좋다.

27.8 배열 메서드

1. 배열에는 **원본 배열(배열 메서드를 호출한 배열, 즉 배열 메서드의 구현체 내부에서 this가 가리키는 객체)을 직접 변경하는 메서드(mutator method)**와,
   원본 배열을 직접 변경하지 않고 새로운 배열을 생성하여 반환하는 메서드가 있다.

27.8.1 Array.isArray

1. 전달된 인수가 배열이면 true, 배열이 아니면 false 반환

27.8.2 Array.prototype.indexOf

1. 원본 배열에서 인수로 전달된 요소를 검색하여 인덱스를 반환
   - 중복되는 요소가 여러 개 있다면 첫 번째로 검색된 요소 인덱스 반환
   - 요소가 존재하지 않으면 -1 반환

2. indexOf 메서드 대신, ES7에서 도입된 Array.prototype.includes 메서드가 가독성이 더 좋다.

27.8.3 Array.prototype.push

1. 전달 받은 모든 값을 마지막 요소로 추가 & 변경된 length 값을 반환

2. 성능 면에서는 좋지 않다.

3. 원본 배열을 직접 변경하기 때문에 push 메서드보다는 ES6의 스프레드 문법 사용을 추천

27.8.4 Array.prototype.pop

1. 원본 배열에서 마지막 요소를 제거 후, 제거한 요소를 반환

27.8.5 Array.prototype.unshift

1. 인수로 전달받은 모든 값을 원본 배열의 선두에 요소로 추가 & 변경된 length 값을 반환

2. 이 방법보다는 ES6 스프레드 문법 추천

27.8.6 Array.prototype.shift

1. 원본 배열에서 첫 번째 요소를 제거하고 제거한 요소 반환
    - 빈 배열이면 undefined 반환

2. shift와 push를 사용하면 큐를 쉽게 구현할 수 있다.

27.8.7 Array.prototype.concat

1. 인수로 전달된 값들을 원본 배열의 마지막 요소로 추가한 새로운 배열을 반환

2. push와 unshift 메서드는 concat 메서드로 대체 가능.
    - 유사하게 동작하지만 **미묘한 차이** 존재
        1. push, unshift 메서드는 원본 배열을 직접 변경, concat은 새로운 배열을 반환
        2. concat 메서드는 인수로 전달받은 배열을 해체하여 새로운 배열의 마지막 요소로 추가

27.8.8 Array.prototype.splice

1. 원본 배열의 중간에 요소를 추가하거나 중간에 있는 요소를 제거하는 경우 사용
    - start : 배열의 요소를 제거하기 시작할 인덱스, start가 -1이면 마지막 요소를 가리킨다.
    - deleteCount : 제거할 요소의 개수다.
    - items : 제거한 위치에 삽입할 요소들의 목록.

27.8.9 Array.prototype.slice

1. 인수로 전달된 범위의 요소들을 복사하여 배열로 반환.
    - start : 복사를 시작할 인덱스
    - end : 복사를 종료할 인덱스

2. 원본은 변경되지 않는다. 이름이 유사한 spilce는 원본 배열을 변경한다.

3. 인수를 모두 생략하면 원본 배열의 복사본을 반환
    - 이때 생성된 복사본은 **얕은 복사**(shallow copy)를 통해 생성

27.8.10 Array.prototype.join

1. 원본 배열의 모든 요소를 문자열로 변환, 인수로 전달받은 문자열, 즉 **구분자**(seperator)로 연결

2. 예시) `join();  join('');`

27.8.11 Array.prototype.reverse

1. 원본 배열의 순서를 반대로 뒤집는다.

27.8.12 Array.prototype.fill

1. 인수로 전달받은 값을 배열의 처음부터 끝까지 채운다.

27.8.13 Array.prototype.includes

1. 배열 내 특정 요소가 포함되어 있는지 확인하여 true / false를 반환

2. indexOf 메서드로는 NaN의 포함 여부를 확인할 수 없다.

27.8.14 Array.prototype.flat

1. 인수로 전달한 깊이만큼 재귀적으로 배열을 평탄화한다.

27.9 배열 고차 함수

27.9.1 Array.prototype.sort

1. 배열의 요소를 정렬한다.
    - 기본적으로 오름차순으로 요소를 정렬

2. 내림차순 정렬 : sort() → reverse()

3. 숫자 요소를 정렬할 때는 주의가 필요.
   - **유니코드 코드 포인트의 순서**를 따른다.

     ➡️ 따라서 숫자요소를 정렬할 때는, 정렬 순서를 정의하는 비교 함수를 인수로 전달해야 함.

27.9.2 Array.prototype.forEach

1. forEach 메서드는 자신의 내부에서 반복문을 실행한다.

2. 콜백 함수를 호출할 때 3개의 인수, 즉 forEach 메서드를 호출한 배열의 요소값과 인덱스, forEach 메서드를 호출한 배열(this)을 순차적으로 전달한다.

3. forEach 메서드는 for문과 달리 break, continue 문을 사용할 수 없다.

4. 희소 배열의 경우 존재하지 않는 요소는 순회 대상에서 제외된다.

27.9.3 Array.prototype.map

1. 자신을 호출한 모든 요소를 순회하면서 인수로 전달받은 콜백 함수를 반복 호출한다.
   그리고 **콜백 함수의 반환값들로 구성된 새로운 배열을 반환**.

2. forEach는 언제나 undefined를 반환, map은 새로운 배열을 반환하는 차이 존재

3. **새로 만들어진 배열은 기존 배열과 1:1 매핑한다.**

27.9.4 Array.prototype.filter

1. 자신을 호출한 배열의 모든 요소를 순회하면서 인수로 전달받은 콜백 함수를 반복 호출한다.
   그리고 **콜백 함수의 반환값이 true인 요소로만 구성된 새로운 배열을 반환.**

27.9.5 Array.prototype.reduce

1. 자신을 호출한 배열의 모든 요소를 순회하면서 인수로 전달받은 콜백 함수를 반복 호출한다.

   그리고 콜백 함수의 반환값을 다음 순회 시에 콜백 함수의 첫 번째 인수로 전달하면서

   콜백 함수를 호출하여 하나의 결과값을 만들어 반환한다.

2. reduce 메서드는 자신을 호출한 배열의 모든 요소를 순회하며 하나의 결과값을 구해야 하는 경우에 사용
   - **평균 구하기**
   - **최대값 구하기**

     : Math.max 메서드를 사용하는 것이 더 직관적.
   - **요소의 중복 횟수 구하기**
   - **중첩 배열 평탄화**

     : flat 메서드가 더 직관적.
   - **중복 요소 제거**

     : filter 메서드가 더 직관적

27.9.6 Array.prototype.some

1. 자신을 호출한 배열의 모든 요소를 순회하면서 인수로 전달받은 콜백 함수를 반복 호출한다.

   콜백 함수의 반환값이 단 한 번이라도 참이면 true, 모두 거짓이면 false 반환.
   - 빈 배열인 경우 언제나 false 반환

27.9.7 Array.prototype.every

1. 자신을 호출한 배열의 모든 요소를 순회하면서 인수로 전달받은 콜백 함수를 반복 호출한다.

   콜백 함수의 반환값이 모두 참이면 true, 단 한 번이라도 거짓이면 false 반환.
   - 빈 배열의 경우 언제나 true 반환

27.9.8 Array.prototype.find

1. 자신을 호출한 배열의 모든 요소를 순회하면서 인수로 전달받은 콜백 함수를 호출하여,

   반환값이 true인 첫 번째 요소를 반환

27.9.9 Array.prototype.findIndex

1. 자신을 호출한 배열의 모든 요소를 순회하면서 인수로 전달받은 콜백 함수를 호출하여, 반환값이 true인 첫 번째 요소의 **인덱스**를 반환
    - 존재하지 않는다면 -1 반환

27.9.10 Array.prototype.flatMap

1. map 메서드를 통해 생성된 새로운 배열을 평탄화한다.
   즉, map 메서드와 flat 메서드를 순차적으로 실행하는 효과가 있다.

2. flatMap 메서드는 1단계만 평탄화하기 때문에,
   평탄화 깊이를 지정해야 하면 map, flat을 각각 호출한다.